

Computational Models - Introduction¹

Handout Mode

Instructors:

Prof. Ronitt Rubinfeld [rubinfeld at post.tau.ac.il](mailto:rubinfeld@post.tau.ac.il)

Dr. Iftach Haitner [iftach.haitner at cs.tau.ac.il](mailto:iftach.haitner@cs.tau.ac.il)

Teaching Assistants:

Dean Doron [deandoron at mail.tau.ac.il](mailto:deandoron@mail.tau.ac.il)

Yuval Moskovitch [mosyuval at gmail.com](mailto:mosyuval@gmail.com)

Oren Salzman [orenzalz at post.tau.ac.il](mailto:orenzalz@post.tau.ac.il)

Tel Aviv University. Spring Semester, 2015. Mondays, 13–16 and Wednesdays, 10–13.

March 9/11, 2015

¹Based on slides by Benny Chor, Tel Aviv University, modifying slides by Maurice Herlihy, Brown University.

Part I

Administrativa

Administrativa

Course website: <http://tau-cm2015b.wikidot.com/>

Site (containing a forum) is our sole mean of disseminating information.

Course Requirements:

- ▶ 6 problem sets (10% of grade).
 - ▶ **Readable, concise, correct** answers expected.
 - ▶ Late submission will **not be accepted**. (You have between one and two weeks, start working when you get them. Any excuse has to cover **all** the period.)
 - ▶ See more instructions on the course website.
 - ▶ Solving problems **independently** is **highly recommended**.
 - ▶ (variants of) questions from HW in exams!

AdministraTrivia II

- ▶ **Midterm**, covering first 6 lectures (up to computability). We will discuss it when we get closer.
- ▶ Midterm is scheduled to **Friday, May 08, 2015**.
- ▶ **Final exam**, covering **all course material**.
- ▶ You must pass the final exam to get a passing course grade.
- ▶ Final Grade: $0.75 * \text{Exam} + 0.15 * \text{Midterm} + 0.10 * \text{HW}$.
- ▶ Prerequisites (formally): Extended introduction to computer science
 - ▶ But most importantly is “mathematical maturity”.
 - ▶ Students from other disciplines with mathematical background encouraged to contact the instructor.
- ▶ Textbook: Sipser — *Introduction to the theory of computation*, first or second editions.
- ▶ Other (excellent) book: Hopcroft, Motwani, and Ullman — *Introduction to Automata Theory, Languages, and Computation*.

Part II

Course overview

Why study theory?

- ▶ Basic computer science issues
 - ▶ What is a computation?
 - ▶ Are computers omnipotent?
 - ▶ What are the fundamental capabilities and limitations of computers?

- ▶ Pragmatic reasons
 - ▶ Avoid intractable or impossible problems.
 - ▶ Apply efficient algorithms when possible.
 - ▶ Learn to tell the difference.

Course Topics

- ▶ **Automata Theory:** Basic model of computation.
 - ▶ Re-invented in many other disciplines.
- ▶ **Computability Theory:** What can computers do?
 - ▶ True impossibility results.
- ▶ **Complexity Theory:** What makes some problems computationally hard and others easy?

Automata theory – Simple models

- ▶ **Finite automata.**
 - ▶ Related to controllers and hardware design.
 - ▶ Useful in text processing and finding patterns in strings.
 - ▶ Probabilistic (Markov) versions useful in modeling various natural phenomena (e.g. speech recognition).
- ▶ **Push down automata.**
 - ▶ Tightly related to a family of languages known as **context free languages**.
 - ▶ Play important role in compilers, design of programming languages, and studies of **natural** languages.

Computability Theory

In the first half of the 20th century, mathematicians such as Kurt Göedel, Alan Turing, and Alonzo Church discovered that some fundamental problems cannot be solved by computers.

- ▶ **Proof verification** of statements can be automated.
- ▶ It is natural to expect that **determining validity** can also be done by a computer.
- ▶ **Theorem:** A computer **cannot determine** if mathematical statement true or false.
- ▶ Results needed theoretical models for computers.
- ▶ These theoretical models helped lead to the construction of real computers.

Halting Problem

Can we decide if a program would halt on a given input?

Answer: No!

- ▶ **Proof idea:** Show a **paradox** assuming such a program exists
- ▶ Assume there exists a program **DHALT(prog,input)** that returns **TRUE** iff **prog** halts when run on **input**.
- ▶ Use **DHALT** to build program **HALT** that on input x :
 - ▶ Executes an infinite loop, if (program) x halts on (input) x .
 - ▶ Halts, otherwise (i.e., if x does **not** halt on x).
- ▶ Is **HALT** implementable?
- ▶ What **HALT(HALT)** does?
- ▶ Hence, **DHALT** does not exist!

Some Other Undecidable Problems

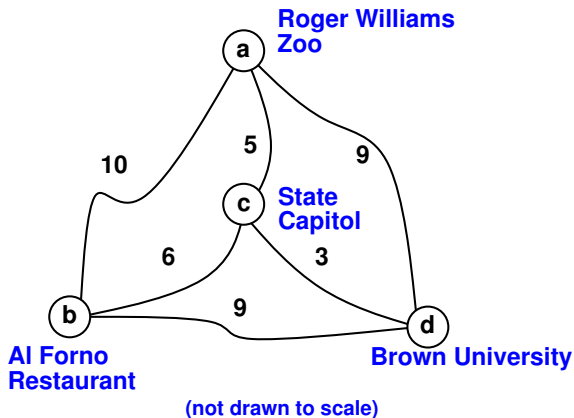
- ▶ Does a program run forever?
- ▶ Is a program correct?
- ▶ Are two programs equivalent?
- ▶ Is a program optimal (time wise – is it the fastest program solving a specific problem)?
- ▶ Does a polynomial equation with one or more variables and integer coefficients (e.g., $5x + 15y + 19xyz^2 = 12$) have an integer solution? (Hilbert's 10'th problem)
- ▶ How compressible is a given string?

Complexity Theory

Key notion: **tractable** vs. **intractable** problems.

- ▶ A **problem** is a general computational question:
 - ▶ description of parameters
 - ▶ description of solution
- ▶ An **algorithm** is a step-by-step procedure
 - ▶ a recipe
 - ▶ a computer program
 - ▶ a mathematical object
- ▶ We want the most **efficient** algorithms
 - ▶ fastest (usually)
 - ▶ most economical with memory (sometimes)
 - ▶ expressed as a function of problem size

Example: Traveling Salesman Problem

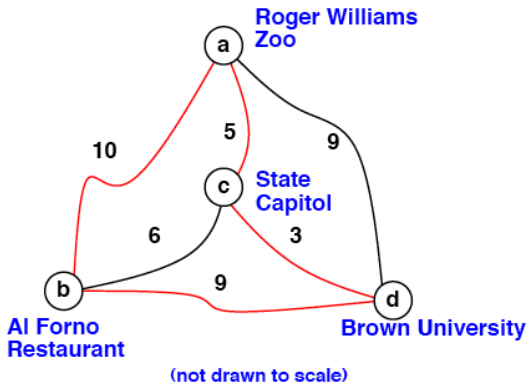


Input:

- ▶ set of **cities**
- ▶ set of inter-city **distances**

Goal: want the shortest tour through the cities

Example: Traveling Salesman Problem



Example: a, b, d, c, a has length 27

Problem Size

- ▶ What is an appropriate measure of problem size?
 - ▶ m nodes?
 - ▶ $m(m+1)/2$ distances?
- ▶ **Encoding** of a problem
 - ▶ alphabet of symbols
 - ▶ strings: $a/b/c/d//10/5/9//6/9//3$.
- ▶ Measures
 - ▶ **Problem Size**: length of encoding (above: 23 ascii characters).
 - ▶ **Time Complexity**: how long an algorithm runs, as function of problem size?

Time Complexity - What is tractable?

- ▶ A function $f(n)$ is $O(g(n))$, if there is a constant c such that for large enough n , $|f(n)| \leq c \cdot |g(n)|$.
- ▶ A **polynomial-time algorithm** is one whose time complexity is $O(p(n))$ for some polynomial $p(n)$, where n denotes the length of the input.
- ▶ An **exponential-time algorithm** is one that runs in time 2^{cn} for some $c > 0$ (thus, not polynomial time).

Part IV

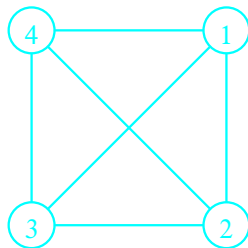
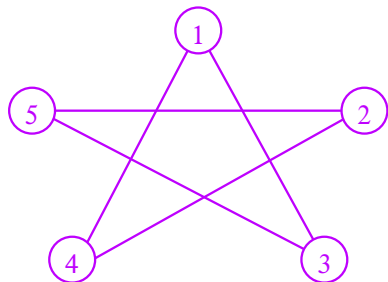
DIY — A (Very) Short Math Review

A Very Short Math Review

- ▶ Graphs
- ▶ Strings and languages
- ▶ Mathematical proofs
- ▶ Mathematical notations (sets, sequences, ...) ✓
- ▶ Functions and predicates ✓

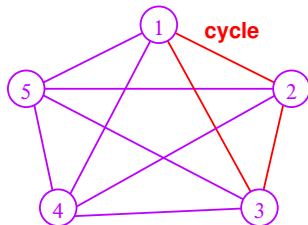
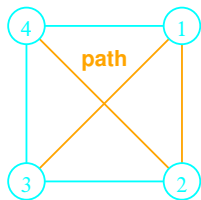
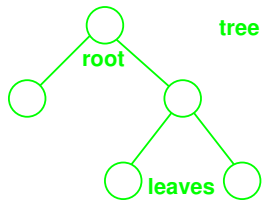
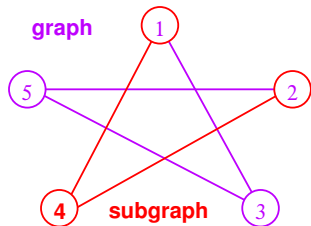
✓ = will be done in recitation.

Graphs

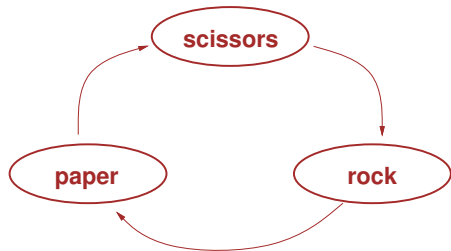
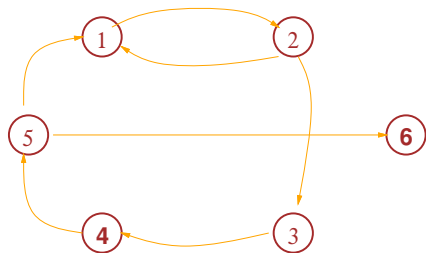


- ▶ $G = (V, E)$, where
 - ▶ V is set of **nodes** or **vertices**, and
 - ▶ E is set of **edges**
 - ▶ **degree** of a vertex is number of edges

Graphs



Directed Graphs



Directed Graph and its Adjacency Matrix

Which **directed** graph is represented by the following 6-by-6 matrix?

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$A(i, j) = 1 \iff (i, j) \in E$$

Strings and Languages

- ▶ an **alphabet** is a finite set of **symbols**
- ▶ a **string over an alphabet** is a finite sequence of symbols from that alphabet.
- ▶ the **length** of a string is the number of symbols
- ▶ the **empty string** ϵ
- ▶ **reverse**: *abcd* reversed is *dcba*.
- ▶ **substring**: *xyz* in *xyzzzy*.
- ▶ **concatenation** of *xyz* and *zy* is *xyzzzy*.
- ▶ x^k is $x \cdots x$, k times.
- ▶ a **language** \mathcal{L} is a (possibly infinite) set of strings.

Proofs

We will use the following basic kinds of proofs.

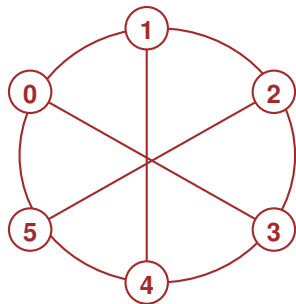
- ▶ by **construction**
- ▶ by **contradiction**
- ▶ by **induction**
- ▶ by **reduction**
- ▶ we will often mix them.

Proof by Construction

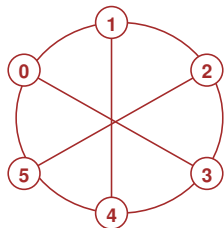
A graph is k -regular if every node has degree k .

Theorem 1

For every even $n > 2$, there exists a 3-regular graph with n nodes.



Proof by Construction



Proof: Construct $G = (V, E)$, where $V = \{0, 1, \dots, n-1\}$ and

$$E = \{\{i, i+1\}: \text{for } 0 \leq i \leq n-2\} \cup \{n-1, 0\} \\ \cup \{\{i, i+n/2\}: \text{for } 0 \leq i \leq n/2-1\}.$$

$$\text{degree}(i) = |\{(i, i+1), (i, i-1), (i, i+n/2)\}| = 3$$

Missing details?

Note: A picture is helpful, but it is **not** a proof!

Proof by Contradiction

Theorem 2

$\sqrt{2}$ is irrational.

Proof: Suppose not, then $\sqrt{2} = \frac{m}{n}$, where m and n are relatively prime.

$$n\sqrt{2} = m \implies 2n^2 = m^2$$

So m^2 is even $\implies m$ is even (?). Let $m = 2k$.

$$2n^2 = (2k)^2 = 4k^2$$

$$\implies n^2 = 2k^2$$

Thus n^2 is even, and so is n . Therefore both m and n are even, and not relatively prime!

Proof by Induction

Prove properties of elements of an infinite set.

$$\mathbb{N} = \{1, 2, 3, \dots\}$$

To prove that φ holds for each element, show:

- ▶ *base step*: show that $\varphi(1)$ is true.
- ▶ *induction step*: show that if $\varphi(j)$ is true (the induction hypothesis), then so is $\varphi(j + 1)$.

Induction Example

Theorem 3

All cows are the same color.

Base step: A single-cow set is definitely the same color.

Induction Step: Assume all sets of i cows are the same color.

Divide the set $\{1, \dots, i+1\}$ into $U = \{1, \dots, i\}$, and $V = \{2, \dots, i+1\}$.

All cows in U are the same color by the induction hypothesis.

All cows in V are the same color by the induction hypothesis.

All cows in $U \cap V$ are the same color by the induction hypothesis.

Hence, all cows are the same color.

Quod Erat Demonstrandum (QED).

Induction Example (cont.)



(cows' images courtesy of www.crawforddirect.com/cows.htm)

Proof by Reduction

We can sometime solve problem A by **reducing** it to problem B , whose solution we already know.

Example: k^{th} element using MAX element

Input: $S = \{7, 12, 3, 15, 9, 5, 19\}$, $k = 2$

Reduction to MAX

- ▶ For $i = 1, \dots, k - 1$ Do:
- ▶ $M = MAX(S)$
- ▶ $S = S \setminus \{M\}$
- ▶ Next i
- ▶ Output $MAX(S)$